Infotek Solutions

Unix Shell Programming

Topics to be covered:		• **Shell Prompt** • **Shell Types** • **Shell Comments** • **Variable Names** • **Defining Variables** • **Accessing Values** • **Readonly command** • **Unsetting Variables** • **Shell Common Variables**

Shell Prompt		The prompt, $, which is called command prompt, is issued by the shell. While the prompt is displayed, you can type a command.
		
		Following is a simple example of **date** command which displays current date and time:
		
		`$date`
		`Thu Jun 25 08:30:19 MST 2009`

Shell Types		In UNIX there are two major types of shells 1. **Bourne shell**, default prompt is the $ character **2. C shell,** default prompt is the % character

Shell Scripts		list of commands, which are listed in the order of execution.
		
		Have comments beginning by pound(#)
		
		Shell script begins with a line like : #!/bin/sh
		
		A script that contains pwd and ls command looks
		`#!/bin/bash`
		`pwd`
		`ls`

Shell
 	
Comments
		You can put your comments in your script as follows –
		
		`#!/bin/bash`
		
		`# Your comment is here`
		`# Script follows here:`
		`pwd`
		`ls`
		
		Now you save the above content and make this script executable as follows –
		
		`$chmod +x test.sh`
		
		Now you have your shell script ready to be executed as follows –
		
		`$./test.sh`
		
		This would produce following result –
		
		`/home/solomon`
		`index.htm  unix-basic_utilities.htm  unix-directories.htm`
		`test.sh    unix-communication.htm    unix-environment.htm`
		
		Note: To execute your any program available in current directory you would execute using **./program_name**
		
		`#!/bin/sh`

		# Comment is here # Script follows here: echo "What is your name?" read PERSON echo "Hello, $PERSON"
Sample script		$./test.sh What is your name? Solomon A Hello, Solomon A $
Variable Names		The name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (_). Variable can't begin with number
		The following examples are valid variable names – _SOLOMON TOKEN_A VAR_1 VAR_2
		Following are the examples of invalid variable names – 2_VAR -VARIABLE VAR1-VAR2 VAR_A!
Defining Variables		Variables are defined as follows – variable_name=variable_value

		For example: `NAME="Solomone A"`

Accessing Values		To access the value stored in a variable, prefix its name with the dollar sign ($) –
		
		For example, following script would access the value of defined variable NAME and would print it on STDOUT –
		
		`#!/bin/sh`
		
		`NAME="Solomon A"`
		`echo $NAME`
		
		This would produce following value –
		
		`Solomon A`

Readonly command		For example, following script would give error while trying to change the value of NAME –
		
		`#!/bin/sh`
		
		`NAME="Solomon A"`
		`readonly NAME`
		`NAME="Qadiri"`
		
		This would produce following result –
		
		`/bin/sh: NAME: This variable is read only.`

Unsetting Variables		`unset variable_name`
		
		Above command would unset the value of a defined variable. Here is a simple example –
		
		`#!/bin/sh`

		NAME="Solomon A" unset NAME echo $NAME
Process ID of Current Shell		For example, the $ character represents the process ID number, or PID, of the current shell: $echo $$ Above command would write PID of the current shell – 29949

Shell Common Variables

Variable	**Description**
$0	The filename of the current script.
$n	These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument (the first argument is $1, the second argument is $2, and so on).
$#	The number of arguments supplied to a script.
$*	All the arguments are double quoted. If a script receives two arguments, $* is equivalent to $1 $2.
$@	All the arguments are individually double quoted. If a script receives two arguments, $@ is equivalent to $1 $2.

$?	The exit status of the last command executed.
$$	The process number of the current shell. For shell scripts, this is the process ID under which they are executing.
$!	The process number of the last background command.